Summon, EBSCO Discovery Service, and Google Scholar: Comparing Search Performance Using User Queries

John Vickery
North Carolina State University Libraries, john_vickery@ncsu.edu

Follow this and additional works at: <https://docs.lib.purdue.edu/charleston>

Part of the [Library and Information Science Commons](#)

An indexed, print copy of the Proceedings is also available for purchase at:

<http://www.thepress.purdue.edu/series/charleston>.

You may also be interested in the new series, Charleston Insights in Library, Archival, and Information Sciences. Find out more at: <http://www.thepress.purdue.edu/series/charleston-insights-library-archival-and-information-sciences>.

John Vickery, "Summon, EBSCO Discovery Service, and Google Scholar: Comparing Search Performance Using User Queries" (2015). *Proceedings of the Charleston Library Conference*.
<http://dx.doi.org/10.5703/1288284316238>

This document has been made available through Purdue e-Pubs, a service of the Purdue University Libraries. Please contact epubs@purdue.edu for additional information.

Summon, EBSCO Discovery Service, and Google Scholar: Comparing Search Performance Using User Queries

John Vickery, Analytics Coordinator, North Carolina State University Libraries

The following is a transcript of a live presentation at the 2015 Charleston Library Conference.

John Vickery: Definitely thanks for coming on a Saturday morning. You know, I want to congratulate us all for sticking around to talk about search performance, and then at the same time maybe get a little worried about us, but mainly I think we should feel excited that we have stuck around. I'm used to Charleston sort of in the small rooms at Francis Marion, you know where we're all kind of in a small little group, so I'm going to try to not wander away from the mic. I usually tend to wander around a little bit, so if you can't hear me, if I kind of step off to the side, start flapping around or something and I'll try to get back to a nice sort of podium stance up here, and I'll also try not to fall off the podium.

So, again, thanks for coming. I'm John Vickery from NC State Libraries and I need to back up. My colleague, Karen Ciccone, who I worked with closely on the project, was unable to make it to the conference. I wish she could but unfortunately she couldn't make it. We were originally scheduled to present this together, so what you see me presenting, we definitely worked together as a team and then that's also part of a larger team effort, so this definitely wasn't just me. I should be up here with several other folks who couldn't make it today.

Today I'm going to talk about a project that a team of us did at NCSU to compare the search performance of Summon, EBSCO's Discovery Service, and Google Scholar. Here's what we will be covering. I definitely plan to leave time for questions, so hopefully we can kind of have a little bit of a discussion afterward or don't feel shy to ask questions at the end or even grab me afterward if you want to just kind of take the discussion offline.

I'll start by giving a little background about the project and our objectives. We'll then get into the

methods of how we coded and analyzed the data, and I'll talk about the idea behind randomization tests and why we used that method so if you pulled up the actual blurb to the talk you saw I mentioned randomization tests, so another thing that I think is kind of exciting for us to be talking about today, and then finally we'll take a peek at the results of our comparison.

So, a minute ago I mentioned that the project was a team effort. It was put together by our web scale discovery product team. Karen and I wrote about the project in an article for *Evidence Based Library and Information Practice*. The article was published I think in February or March of this past year. It's open access, so Google it, pull it up, you can read it now while I'm giving you the talk. And if you really want to kind of take a deep dive into some of the idea of randomization tests a little bit more and how we implemented them using SAS, you can take a look at an article I did for the SAS Global Forum this past spring if you follow that link, or maybe Google the conference proceedings and then if you want to maybe implement the code yourself or really start playing around with it, the code is in the conference proceedings paper as well but you can check out the GitHub space and download the code and play around with it. I should say I program in SAS, but I know you could implement the test we did in Aura or Python, or pretty much any analytics language of your choice, so what I'll be talking about is how I did it in SAS. I'm not going to get into the code and whatnot, but that is SAS code on GitHub and in the conference proceedings, but again you could implement this in several different ways. And again, if you decide to take a look at the code and you have questions, please get in touch with me. I'd be glad to talk about it more.

Okay, so a little bit about the project. Similar to many libraries, at NCSU we've invested quite a lot of time, effort, and money into implementing a web scale discovery service. So, we think it's

important to periodically review the landscape, review competing products, and see if maybe it would be a better fit for us because of price or effectiveness or any other consideration so our web scale discovery product team is charged with testing and evaluating our existing service as well as other potential products. Our team is made up of about nine librarians. Some of them may be in the room now but not to worry, you can flag yourself down, and we covered several departments in the libraries. We tried to touch all of them. We range from public services, tech services, collections, and IT, and at NCSU we've been using Summon since 2009.

Something a little different about how we use our discovery service—we primarily use Summon to populate the articles section of our quick search application. So, if you could kind of see that. I've been in a couple presentations here and screenshots tend to get kind of washed out. But, if you are a user at NCSU's library website, we present folks with sort of the large single search box. If you were to enter a search in there, that search is spread out into both our website, article results, and book results, so what we're using our discovery service for is to populate this article section, these article results. So, we'll give you the top three and if you click "see all results" you'll be passed into the Summon interface. Most of our users don't actually even get to the Summon interface, and this has implications in how we designed the test as well, but back when we implemented Summon in 2009, it was actually I think the only discovery service that had an API that we could leverage for this purpose. So, 2009 was a long time ago, and since of course other services have the API functionality, in particular EBSCO's that we wanted to evaluate.

So, in April and May of 2014, we set up a trial of EDS, and it was during that time that we ran the tests. So we were testing in the spring of 2014 and did data analysis over the summer. Of course there has been other really good studies that compared discovery services. I would encourage you to take a look at these two from 2013. Asher, Duke, and Wilson compared the performance of Summon, EDS, and Google Scholar, and their

study quality was judged on whether or not articles selected by test subjects were scholarly publications, so peer-reviewed articles. Rochkind, if I'm pronouncing that right, published a study that compared user performance for search results from several of the available discovery services. In that test subjects were presented with a side-by-side view of two discovery services, and they were asked to select which results set they preferred. Both of those studies had test subjects enter hypothetical searches, and so the main difference between these two and our study and where we think our study contributes to the literature and to the community is that we used actual search queries from our Summon logs. So, we actually found the actual text that was entered into a search and used that as the basis of our search as opposed to asking users to enter in, maybe pick some searches that you think you might want to do.

The primary objective of our study was to answer the question of which discovery service produced better results for the type of searches typically performed by our users. And we can get a sense of what "typical" is by examining a Summon search log. So what I was mentioning before as opposed to hypothetical kind of "try to come up with a question on your own," we were actually drawing the real researcher questions that our users, faculty, students, staff, we didn't know who they were, were entering into a search. So, the Summon logs contained the actual queries that the users entered, and from looking at the logs we know that about three-quarters of the searches are for topical-type sort of broad searches where you're looking for a general area, and about 25% are known-item type searches, so this would be where a user actually pastes in a citation to an article so they know exactly what they were looking for and they're going to go get it. Or maybe they typed in a title keyword or a piece of the title of the work. So, because of how these two searches differ, we wanted to separately evaluate how the discovery services handled them. So, here's a couple examples. This shouldn't look particularly surprising to anybody but you can see the difference. These are actual searches from our logs of known-item search queries on

the left and topical search queries on the right. So, when we're looking at our search logs, this is the kind of how we're going to see the difference in those. We would've had again about 75% of the searches be topical and 25% be known-item, so you can see where someone has actually pasted in the citation to that internal medicine article there.

In the title of the presentation we talked about Google Scholar. So far we have kind of just touched on Summon and EDS, but we needed to consider our 800-pound Google gorilla. So, why did we throw that into the mix? So, first I should say that discovery services like Summon and EDS, they offer clear benefits to libraries that Google Scholar doesn't. In particular, Google Scholar's terms of service—they don't allow results to be presented outside of the Google Scholar context. So, at NCSU where we use the API functionality to populate that articles section of our quick search app, we couldn't do that even if we decided we wanted to sort of say, "okay, we want to use Google Scholar." We can't use it in the way that we have our environment set up now. So, we also know that the Google universe is sort of the "go to" place for many of our researchers, and we could maybe say most, I'm not sure. So we really wanted to see how that fared as well.

Okay, now we're going to start getting down into the weeds a little bit with the methods. This is kind of the part where I get the most interested and sort of my interest is usually inversely related to other folks' interest, so a little "nerd alert." So, hopefully this is the kind of stuff you might want to talk to me about later or just kind of hang on and we can get through it. Okay, so let's talk about how we coded and analyzed the data. The first and probably the most important thing that we did again was to use those actual user search queries from our Summon search logs. So we wanted to try to replicate the experience, really see what our users saw as closely as possible to really replicate what results they were getting. Now it's not uncommon for these searches to have punctuation errors because people probably type better than me, but you're going to get extraneous characters, sort of overly broad topics, but we kept those exactly as they appeared, so if

someone had mistyped something, we just copied and pasted that query into our test to see how it worked. So, a few examples. You can see where there is a "G" in front of plants or some misspellings or "new class of drugs patent," maybe something that doesn't necessarily come off as a well-thought-out topic, but it could've just been mistakes in the typing or how the student or faculty framed the search.

Another really important aspect of the methods was that we generated a random sample from the search logs. I know we all know that this is basic standard practice, but it was really important for us to get a representative sample or a representative dataset to test. So, we started with about a year's—I think a complete year's—worth of Summon's search logs, and we generated a simple random sample so no stratification, just a random sample of 225 queries. The 225 number we felt like was large enough to be representative, but it also worked well with our team of nine members because we gave each team member 25 queries to test. A couple team members were not able to actually complete the test, so our final analysis dataset was down some from 225, but it still after sort of some post-talk power tests came out sufficient to detect an effect that we wanted to detect. So, team members, the nine of us, we used our judgment to classify each query as either topical or a known-item search, and going back to that slide I showed a minute ago, it was generally very easy. In our discussions amongst the team we didn't have a whole lot of talk at all about "I couldn't decide, was this topical or not?" It really was quite clear so that was our first decision point was "Is this query a topical search or a known-item search?" And we're going to split our analysis based on that. For topical searches we recorded the number of relevant results within the first 10 results, and we used our judgment again to consider a result relevant if it matched the presumed topic based on kind of title and abstract only. So, we didn't really dig deep into the result list. We tried to—I guess in a lot of ways we were trying to put ourselves in the frame of a user, a student maybe, and say if you are scanning this result list you're not going to necessarily read

every article. You're going to make the decision at that time: "Is this relevant? Do I want to dive into it more or pass?" For known-item-type searches we asked two questions. One was, "Did you find the item and was it in the top three results?" We know from some of our testing that it's relatively uncommon for folks to actually get past those top three. We limited ourselves to the top 10. I think in a future test, if you wanted to really get even more strict, you could say limit that the entire test environment to just the top three results. Since we used a random sample, the distribution of the topical to known-item that mirrored the overall distribution, so again we within our sample dataset of 225 we had about 75% of those were topical questions and 25% were known-item. And at this point we've got the dataset up for two different types of analysis. For the topical searches we have count data, and for the known-item searches we have binary sort of "yes"/"no" categorical data.

After all the team members coded their data, we had a series of spreadsheets that looked like this. These are examples of the known-item searches with the "yes"/"no." Team members—this is sort of half the spreadsheet. The way we presented it to our team members was a spreadsheet with a list of their 25 queries to test, they needed to make that first judgment of topical or known-item, and then they would respond in the various columns depending on how they determined it whether it was known-item or not. And this would be how they recorded the topical or broad searches, what we also called it. So, for the topical searches, the way the test was designed we had the dataset up appropriately for a repeated measures ANOVA, so analysis of variance, and then what that test is designed to detect is any overall difference in the average number of relevant results for each of the three products. So in a repeated measures ANOVA, we test each subject multiple times, once under each condition. And let me show you how we were viewing this as that set up. So you can think of the query text as the subject and condition one will be testing under EDS, condition two under Summon, and condition three under Google Scholar. So, kind of in general you may see this kind of design

in like a medical study or something where one individual was given three different drugs over different periods of time, but it was the same individual that was being tested, so in a sense the query text is the individual and the discovery service is the condition that we tested under.

So, we pulled the data into SAS, and again like I said you could use other analytic tools. SAS is the one I tend to use, and the first stop was to make some plots or some graphs of the data and get some basic stats. I'm going to kind of jump ahead a little bit and show you the plot of the known-item searches first, and when I saw it, it was kind of a little bit of a spoiler for the results section because it was a pretty boring graph. It was like a tie. So, in some sense, we didn't always need to run complicated analysis. After kind of viewing this we thought, "Well, that was kind of interesting. It's a tie." I'll make a comment though about the actual dataset of 44, that got a little too small after some of the folks had to drop out of the test, so sort of in a follow-up this would be an area that would really want to address as to kind of take a deeper dive into the known-item type queries. We had a large enough dataset for the topical searches, but this was kind of pushing the boundary on known-items of this would be a spot to, sort of the weakness of the study and where we would really want to evaluate it more in-depth. And then for the other question of was it in the top three results? All but one of the queries was in the top three for Summon, and all but two for EDS and Google Scholar. So, again, really close basically in some senses a tie for the known-item type searches.

So, let's go back to the topical searches. These are the average number of relevant results for each of the three discovery services. Summon and EDS are basically tied. You can see 4.76 and 4.83. That's really close, and then you can see that Google Scholar had about one additional relevant result per result set within the first 10. So, overall these are coming in really, really close. When we plotted it out you can see the distribution of the number of relevant results so you see EDS is blue, Google is red, and Summon is green. You can see where they had the frequency of 0 relevant results, 10

relevant results, and so we've got the distribution from 0 to 10, and you can see that there is enough variation to kind of warrant a closer look, so different from the known-item we're kind of seeing a little more variation where we'll want to kind of dive a little more deeper into it. But at the same time this is where I started to get nervous about using a regular repeated measures ANOVA, because like all statistical tests, there's assumptions that you have to meet and one of them you know you can breed, whether or not this is one you need to worry about all the time or not, but one of them is that the data should be normally distributed. Get you a nice bell curve and this is really, really, really not a nice bell curve. In my analysis work I typically don't get a nice bell curve. One day I feel like I'm going to find one but in library data I never do. Ever. But one day maybe I'll just get a nice normal distribution. So, in order to get around the possible violations of the assumptions that are regular repeated measures of ANOVA, we decided to use what's called a randomization test. Now the concept could be maybe a little complex, but I kind of find that randomization tests are in a way easier to conceptualize than the normal parametric equivalent. So, what in the world is a randomization test? The concept goes back to the 30s with R. A. Fischer. This is a picture of him and others and the basic outline of the randomization test—you might see it called randomization tests or permutation test—is that you randomly shuffle the data and repeatedly calculate your test statistic and the “P” value, that marker of statistical significance, where you want it to be under .05 or .001 is the fraction of how many times the test statistic for the shuffled data, that rearranged data, is equal to or more extreme than the observed test statistic. All right. If you're still awake, give yourself a cheer. You just made it!

Okay, now the practicality of the method, however, was limited by computing power until relatively recently because even with small datasets the number of possible permutation gets really, really big really fast, so our dataset had 139 observations, so 139 rows, and the way that we're permeating the data was within that row.

So we would take an answer from EDS and put it in the Summons slot. This is randomly generated so or the number recorded for Google Scholar and slot that into the EDS slot. So the formula gives you the possible number of permutations, but that is six to the 139th power. You know, I put that in and tried to like put all the zeroes out there. It's a really huge, huge number. So, that's why they weren't doing it in the 30s, because they couldn't write fast enough. But, you don't actually have to do it that many times. Typically you'll see randomization tests done in the low thousands or ten thousands, which for your computer is nothing. You can run a randomization test and do ten thousand iterations of it and it's really quick.

The basic steps: you calculate your test statistic. In our case we were doing ANOVA, so we chose the “F” statistic—that's the standard one for ANOVA. You rearrange the data like I was saying before. You're going to shuffle it up and then you're going to calculate the test statistic for this shuffled up data. Then you're going to do number two and three many times. I heard a presentation the other day where someone was referring to this as like, “wash, rinse, repeat,” because this is kind of where you're reading the back of the shampoo bottle. You're doing number two and number three many times, but your computer is actually doing it, and then your “P” value is the fraction of how many times the statistic for the rearranged data, all of it put together, is equal to or more extreme than the observed test statistic.

Okay, so let's look at the results. You've kind of suffered through the methods. We saw that the known-item searches were basically a tie but the topical searches where we applied these randomization tests. Okay, histograms I love. Histograms I think they're fun to look at. Hopefully you do too. With the topical searches our randomization test showed that there was an overall difference. If you think back to that table where I showed you the average results, we had Summon and EDS as a tie, and Google just about won. But we are looking for is whether or not that slight difference was due to chance or whether or not there was an actual overall difference. So, here's what it looks like plotted out the results of

our randomization test. And what you want to notice about this plot is the area of the graph to the right of the redline. It's that little bitty shrunk down spot. So, that's sort of the graphical representation of the "P" value. That's our "P" value of .002, which is a really, that's a small number, and so that's strong support for an overall difference. Basically what we're seeing is that it's really unlikely that we would've gotten our observed difference, our observed "F" value of 6.3, just by random chance alone. So, were saying to ourselves, "Wow, there really is a difference in these three products!" Now again the difference was slight, but there really is a difference. It wasn't just chance. Now, our ANOVA test or our randomization test shows that there is an overall difference, but it doesn't show us where. So, we don't know which discovery service is accounting for the difference. I mean, we can have an idea of that based on our table of looking at the average number, but in order to figure it out we had to do pair wise comparisons, so we did more randomization tests. This time comparing the possible pairs. So, Summon to EDS, EDS to Google Scholar, and Summon to Google Scholar. Here is what we got comparing Summon to EDS. What you want to take away from this graph as opposed to the previous one is the large "P" value and the shaded red areas. What this tells us is that there is essentially no difference in the way we measured relevant results in our test between Summon and EDS. So if we have Summon and EDS folks here you're basically tied according to the way we tested it. So, I don't know, you can take it out into the hall and see who's really winning. Now, I think of a lot of times when you read studies you get excited for a difference. We were excited to see a difference based on the "F" test that we did a minute ago, but this sort of no difference, this tie for us at NCSU had really important implications. It actually was really in a sense very helpful because it meant that we could basically take performance out of the evaluation equation. So when our team is looking at which product to choose and what factors to weigh we can say to ourselves "performance isn't one of those," and in our case and the way we tested it, performance isn't in the equation, so we can focus on backend functionality, the knowledge-based

piece that comes with the discovery services. We can focus on user interface so our user research team might want to go out and do a test and just focus on which interface works best for our users or we can look at cost, so it's basically just simplifying our decision making, which is really important and really helpful for us at NCSU.

Okay, so what about Google Scholar? This is Summon compared to Google Scholar, and you'll notice a difference here as opposed to a lot of red if you can actually see what you want to focus on again is the area to the right of the red lines, and just a very little red shaded area, and that's the graphical representation of our really small "P" value. So again this is saying that Summon in fact according to our test did significantly, you know, in a statistical sense, outperform Summon. So Google really did have better results. I'm sorry; this is showing that Google Scholar did in fact, thank you, outperform Summon. Statistically. Okay, and so now we've got to compare EDS to Google, and no surprise we basically—another version of that previous histogram. Again we saw a slight difference with Google Scholar outperforming by almost one relevant result, and our randomization test showed that that was super, super, super unlikely to have been just based on random chance alone. So again Google Scholar did outperform it as the way we tested it, both Summon and EDS. And again this has important implications for us at NCSU. On the one hand we are not able to plug Google Scholar into our infrastructure the way we've designed it, the way we want to use it, but we can also say to ourselves and I guess that our users are actually getting good relevant results when they choose to go to Google Scholar, which we know they are. So, basically the same story between EDS and Summon—Google Scholar did actually outperform them, but do recall that it was a relatively slight difference.

So, a few takeaways. Summon and EDS basically tied, so they're going to have to duke it out in areas outside of performance, or maybe that will change if we redo the test at a future point or or if any of you guys decide to you might see a difference in their performance, but for us at

NCSU it took performance out of our evaluation equation. It's simplified our decision making. We were able to focus on cost, backend performance, and user interface. Google Scholar came out slightly ahead like we just saw. Again it's kind of important for our users because we know that is what they're using, and this is my little takeaway hopefully that you guys might want to consider a

randomization test as something in your toolbox of hypothesis testing. It might be an option for your data because again in my experience I typically don't get nice, normal data, so it's been helpful to have a sort of a more robust, nonparametric test that we can turn to. Thank you very much. I'm definitely happy to answer questions if you have them.